



IBM Informix® Dynamic Server™ (IDS)  
IDS Problem Determination Tutorial Series

---

## Combining IDS and OS Diagnostics

## Table of Contents

About this tutorial .....	3
Introduction.....	3
Setup .....	4
Tutorial Conventions Used .....	4
About the authors.....	4
<i>Section 1 Understanding the Environment</i> .....	6
Section 1.1 Introduction to the Operating System (OS) Environment .....	6
Exercise: Create and Run the idssupport Command to Obtain OS Environment .....	6
Section 1.2 Introduction to the Engine Environment .....	9
Section 1.3 The Middleware Environment .....	9
Section 1.4 The End User Environment: .....	10
Section 1.5 Relevant IDS Utilities and Files .....	10
Section 1.6 Sources of Information for IDS .....	11
Section 1.7 Specific OS Platform Commands .....	11
<i>Section 2 Diagnostic and Event Logs</i> .....	14
Section 2.1 Correlating IDS and System Events or Errors .....	14
Section 2.2 Windows Event Logs and the Dr. Watson Log .....	16
Section 2.3 Avoiding Pitfalls and a First Glance at System Data.....	16
Section 2.4 Using perfmon on Windows .....	18
<i>Section 3 Operating System Error Examples on UNIX</i> .....	22
Section 3.1 File Permissions and Space Issues .....	22
Section 3.2 Connectivity Errors: 951, 952, 956 25588, 25582 .....	24
Section 3.3 Shared Memory Related Issues.....	27
<i>Section 4 Defining a Performance Problem and Initial Data Collection</i> .....	31
Section 4.1 Clarifying the Problem.....	31
Section 4.2 IDS diagnostics that can be performed quickly .....	32
Section 4.3 IDS function call stacks, trap files, and tracing .....	34
Section 4.4 OS tracing and debugging utilities.....	38
Summary and feedback .....	39
What you should know .....	39

## Combining IDS and OS Diagnostics

### ***About this tutorial***

#### **Introduction**

The objectives of this tutorial are to guide you through the initial steps in defining and diagnosing system errors that involve IDS. Specifically, you will learn the following:

- What information to collect from IDS tools and logs
- What information to collect from operating system tools and logs
- What information to collect or look for from Middleware/Client tools and logs
- What environmental information to collect
- How to use all this information together in problem investigation.

The topics in this tutorial include understanding configuration, connection issues, shared memory utilization, query performance analysis and basic diagnostic log analysis. The following platforms are covered:

- AIX
- Solaris
- HP-UX
- Linux
- Windows

This tutorial is for anybody who reports or analyzes IDS problems where at least one of these conditions applies:

- The operating system or user environment may be a factor
- There is an issue with a resource managed by the operating system, such as CPU, memory, interprocess communication (IPC) resources, or disks

You should already have the following skills:

- Are familiar with the basic concepts of an operating system - including RAM, CPU, virtual memory, IPC resources, and I/O
- Are familiar with basic operating system commands
- Understand IDS concepts including Shared Memory Classes, VPs, TCP protocol, Buffers, extents, fragmentation, PDQ parallelism, IDS process/thread names

Prior to taking this tutorial, it is suggested that you review the following chapters from the IDS 9.3 *Administrator's Guide*:

Chapter 2 – Configuration Parameters

Chapter 3 – Client/Server Communications

Section II – Disk, Memory and Process Management

From the IDS 9.3 *Administrator's Reference* :

Chapter 1 – Configuration Parameters

Chapter 2 – The Sysmaster Database

Chapter 3 – Utilities

Chapter 5 – Disk Structures and Storage

Appendix A – Files That the Database Server Uses

From the IDS 9.3 *Performance Guide*:

Chapter 1 – Performance Basics

Chapter 2 – Performance Monitoring

Chapter 3 – Effects of Configuration on CPU Utilization

Chapter 4 – Effects of Configuration on Memory Utilization

Chapter 5 – Effects of Configuration on I/O Utilization

Chapter 13 – Improving Individual Query Performance

These guides can be viewed online at the IBM Informix Technical Support Web site:

[http://www-3.ibm.com/software/data/informix/pubs/library/ids\\_93.html](http://www-3.ibm.com/software/data/informix/pubs/library/ids_93.html)

## Setup

IDS version 9.3 or greater should be installed on the platforms you have available. Some examples illustrate platform-specific behavior or tools on AIX, Linux, Solaris, and Windows. It is possible to work through some of the UNIX examples on any of the supported UNIX platforms, though specific commands and their output will differ.

The `stores_demo` database should be created with the **dbaccessdemo9** command.

You should have administrator or root privileges on the systems where you are working. If administrator privileges cannot be obtained, there are a few exercises that you might not be able to complete.

You should also note that some examples in this tutorial will create stress on the system, so other users may be affected. The system may also become unstable.

## Tutorial Conventions Used

When a tool or utility is first mentioned it will be shown in **bold** text.

All command statements and their output will be shown in a `monospaced` font.

Some examples will show specific command options which may change over time, which will always be documented in IDS documentation.

## About the authors

Selvaprabhu Arumugharaj is an advanced technical support specialist with the IDS Support Team. He specializes in down systems and diagnostics work.

Don Hutchinson is a technical support specialist with the IDS Support Team. He specializes in escalations that involve solving problems which cross the boundary between the IBM IDS engine and operating systems.

Jeff Ousley is a technical support specialist with the IDS Support Team. He specializes in escalations that involve datablades, the datablade API and extensibility areas of the IBM IDS engine.

Carl Williams is a technical support specialist with the IDS Support Team. He specializes in escalations that involve Enterprise Gateway Manager, DRDA and Assert Failure diagnostic areas of the IBM IDS engine.

Jim Wright is a technical support specialist with the IDS Support Team. He specializes in performance tuning and data warehousing areas of the IBM Informix product lines.

You can reach these authors by locating their e-mail address in the IBM Global Directory at <http://www.ibm.com/contact/employees/us> .

## ***Section 1 Understanding the Environment***

### **Section 1.1 Introduction to the Operating System (OS) Environment**

Diagnosing some problems related to memory, swap files, CPU, disk storage, and other resources requires a thorough understanding of how a given operating system manages these resources. At a minimum, defining resource-related problems requires knowing how much of that resource exists, and what resource limits may exist per user. (The relevant limits are typically for the user ID of the IDS instance, which is either root or informix.)

Here is some of the important configuration information that you may need to obtain:

- Operating system patch level, installed software, and upgrade history
- Number of CPUs
- Amount of RAM
- Swap and file cache settings
- User data and file resource limits and per user process limit
- IPC resource limits (message queues, shared memory segments, semaphores)
- Type of disk storage (for example EMC, Shark, Network Access Storage solution)
- What else is the machine used for? Does IDS compete for resources?
- Where does authentication occur?

Most platforms have straightforward commands for retrieving resource information. By placing them in a simple shell script, you can execute an information gathering process with one command. And you can tailor this to your needs at any given time. This exercise shows how to create a bourne shell script to gather the output from some basic commands.

### **Exercise: Create and Run the idssupport Command to Obtain OS Environment**

1. On each platform that you have available, start IDS with the **oninit** command.
2. Assuming you already have the stores\_demo database available, create a directory for storing the output from idssupport.
3. Change to that directory and issue:

```
vi idssupport
```

4. Enter these lines:

```
#!/bin/sh
#- Example of an information gathering script

#- Get OS name -#
```

```
osname=`uname`

PATH=${PATH}:/usr/bin:/usr/sbin
export PATH

#- Setup up log file -#
case "$osname" in
SunOS)  myid=`/usr/xpg4/bin/id -un`
;;
*)      myid=`id -un`
;;
esac
LOG="./`basename $0`. $myid"

echo "== $LOG == Taken at: `date` ==" >$LOG
case "$myid" in
root)
echo " -- OS Information --" >>$LOG
case "$osname" in
AIX)
sarropts="-P ALL"
ioopts=""
oslevel -r >>$LOG
lsdev -C -c memory >>$LOG
lsattr -El sys0 >>$LOG
lslpp -ah >>$LOG
instfix -i |grep ML >>$LOG
cat /etc/security/limits >>$LOG
errpt -a >>$LOG
/usr/samples/kernel/vmtune >>$LOG
;;
HP-UX)
sarropts="-A"
ioopts="-xt"
ioscan >>$LOG
dmesg >>$LOG
kmtune >>$LOG
echo "getconf KERNEL_BITS:" >>$LOG
getconf KERNEL_BITS >>$LOG
swlist -v >>$LOG
tail -50 /var/adm/syslog/syslog.log >>$LOG
;;
SunOS)
sarropts="-A"
ioopts="-xcn"
psrinfo -v >>$LOG
prtconf >>$LOG
swap -l >>$LOG
sysdef -i >>$LOG
patchadd -p >>$LOG
showrev -p >>$LOG
tail -50 /var/adm/messages >>$LOG
```

```
tail -50 /var/log/syslog >>$LOG
prtconf -v >>$LOG
;;
Linux)
    saropts="-A"
    ioopts="-t"
    dmesg >>$LOG
    swapon -s >>$LOG
    rpm -qa >>$LOG
    tail -50 /var/log/messages >>$LOG
;;
esac

#- All OS's -#
iostat $ioopts 1 2 >>$LOG
sar $saropts 1 2 >>$LOG
vmstat 1 2 >>$LOG
df -k >>$LOG
uname -a >>$LOG
netstat -a >>$LOG
ulimit -a >>$LOG
set >>$LOG
env >>$LOG
ps -ef|grep oninit >>$LOG
;;
informix)    #- Engine
    echo " -- Engine Information --" >>$LOG
    env >>$LOG
    ulimit -a >>$LOG
    #- These vars must be set for onstats to work:
    #-   INFORMIXDIR, INFORMIXSERVER, ONCONFIG, PATH
    onstat -c >>$LOG
    echo " -- Onstats --" >>$LOG
    onstat -g ses >>$LOG
    onstat -g seg >>$LOG
    onstat -m >>$LOG
;;
middlewareuser)    #- Middleware user
    echo " -- Middleware Information --" >>$LOG
    echo " " >>$LOG
;;
*)    #- End user
    echo " -- User Information --" >>$LOG
    env >>$LOG
    ulimit -a >>$LOG
;;
esac

echo "== End of $LOG ==" >>$LOG

echo "$0 report ran as $myid complete."
echo " See file $LOG for output."
```

```
echo " "
```

5. Set the permission on the file to execute:  
`chmod 744 idssupport`
6. Run the script as user root:  
`./idssupport`
7. Analyze your output and find the following items:
  - Number of CPU's
  - Memory Total, Virtual and current usage
  - System Log output
  - Disk usage
  - OS patches installed

## Section 1.2 Introduction to the Engine Environment

You have at least 2 environments present in any IDS connection. You can have the same values but these environments are distinct. The engine or IDS environment is the starting point for all connections. The IDS environment consists of your version, ONCONFIG, environment variables and the OS environment. You can modify this environment.

### EXERCISE: Getting the Engine environment:

1. Login as user informix. Setup the engine environment and start the engine. Then type:  
`./idssupport`
2. Locate the idssupport.informix file.
3. Analyze this output and find the following:
  - The total number of sessions
  - The maximum file size for the user
  - The database server name used to connect to the engine
  - The environment variables that have the INFORMIXDIR path in them.
  - The size of memory the database server is using
  - The number of user sessions connected to the engine

## Section 1.3 The Middleware Environment

An additional environment can be present if you have a middleware product such as ODBC, JDBC, or 3rd party application. For each type of middleware there is a global starting point you can tailor to your needs. This works similar to your user environment in that it starts with the connection environment and adds to it. The connections may have their own environment. Any diagnostics captured must also include this layer or the full picture will not be known.

### EXERCISE: Getting the Middleware (if present) environment:

1. If you have a middleware product and that product runs processes as a specific user, edit the idssupport script and find the section called middlewareuser.

```
vi idssupport
```

```
middlewareuser)#- Middleware user
echo " -- Middleware Information --" >>$LOG

env >>$LOG
ulimit -a >>$LOG
echo " " >>$LOG
;;
```

2. Change the middlewareuser tag to the name of the middleware product userid and add commands that the middleware provides to gather diagnostics.
3. Save your changes.
4. Login as the middleware user and run the script: ./idssupport
5. Locate and analyze the idssupport file. Note the values for the middleware environment variable you asked to gather.

## Section 1.4 The End User Environment:

Your user environment consists of the default engine session variables, a subset of the engine environment and any overrides you set in the environment or in a connection startup file. You also have a client version to consider. The lack of a variable generally means it is set to a default, but it does exist.

### EXERCISE: Getting the End user environment:

1. Login as the end user and run the idssupport script:  
./idssupport
2. Analyze the idssupport file. Find the following:

The environment variables that have the INFORMIXDIR value set  
The maximum file size the user can create  
The database server name that the user connects to.

## Section 1.5 Relevant IDS Utilities and Files

The most important IDS utility for OS and IDS problem determination is the onstat utility. The Administrator's Reference manual covers the options, but there are many undocumented ones that are available if they are needed. You should not be concerned if Technical Support advises a command that is not in the manual.

The most important IDS files are:

The \$INFORMIXDIR/etc/\$ONCONFIG file  
The ONCONFIG:MSGPATH file commonly called the online.log  
The sqlhosts or \$INFORMIXSQLHOSTS

You can solve most problems reported by the OS or IDS with a thorough knowledge of these utilities and files.

IDS uses many files for its operations that are documented and undocumented. You should be familiar with the documented ones. They are listed with a brief description in The Administrator's Reference Appendix A. The key files are:

- /INFORMIXTMP/\*
- \$INFORMIXDIR/etc/oncfg\_.
- Root dbspace reserved pages.
- Connectivity files used in user authentication. /etc/passwd, /etc/hosts, /etc/services, /etc/nsswitch.conf, etc.
- The dbspace paths of the chunks used by IDS.

You should understand the location and be able to verify the contents of these files in order to troubleshoot OS and IDS issues.

## Section 1.6 Sources of Information for IDS

When a problem occurs, the first thing that will be checked by Support is whether the engine is set to the recommendations in the release notes. You should review these with each release and periodically check the IBM knowledge centers for the latest information.

### EXERCISE: Release Notes

1. Login to your instance and find the release notes for the server.  
`cd $INFORMIXDIR/release/en_us/0333`
2. List the directory contents. Note the convention of the naming and version.
3. Find the `ids_machine_notes_9.30.txt` file. You will find that these names change quite often. Older servers release notes were simply `IDS_7.3`. So read these and ensure your machine is set or has the minimum patches listed in the machine release notes.
4. Locate the reference to the SHMBASE setting for your platform and verify your \$ONCONFIG has the correct setting.

### EXERCISE: Search the IBM Knowledge Centers

1. Open a web browser and go to <http://www.ibm.com/software/support>
2. Type in the search text: informix onbar
3. Click the Solve a Problem (FAQ's) button
4. If you have access open one of the documents that has a key. If not, scroll down until you find an article that does not require support.

## Section 1.7 Specific OS Platform Commands

The following chart lists the equivalent commands for each Operating System. For details and options, refer to that Operating System's Administration manuals and man pages.

# IDS Problem Determination Tutorial Series

## Combining IDS and OS Diagnostics

Platform	AIX	HP-UX	Solaris	Linux	Windows
<b>OS Level</b>	/usr/sbin/oslevel	/usr/bin/uname -a	/usr/bin/uname -a	/bin/uname -a cat /etc/redhat-release rpm -qa   grep -i kernel rpm -qa   grep -i glibc	winmsd /af (NT) creates a file: .TXT C:\Program Files\Common Files\Microsoft\Shared\MsInfo\Msinfo32\report (Windows 2000® / XP®)
<b>CPU Defined</b>	/usr/sbin/lssdev -C \  grep proc	/usr/sbin/ioscan	/usr/sbin/psrinfo -v	/bin/dmesg cat /proc/cpuinfo	winmsd (Processor) msinfo32 System Summary
<b>Memory Limits</b>	/usr/sbin/lssattr -E \ sys0 /usr/samples/kernel /vmtune	/usr/sbin/dmesg	/usr/sbin/prtconf swap -l ps -eal (per process)	/bin/dmesg /usr/bin/free cat /proc/meminfo /usr/bin/procinfo	winmsd Memory Report msinfo32 System Summary Memory
<b>IPC Limits</b>	N/A	/usr/sbin/kmtune getconf	/usr/sbin/sysdef -i /etc \ /system/etc/sysdef	/usr/bin/ipcs -l	N/A
<b>Software Level</b>	/usr/bin/lslpp -ah /usr/bin/lslpp -L instfix -i  grep ML	/usr/sbin/swlist -v	/usr/sbin/patchadd -p /usr/bin/showrev -p \ grep \ SUNW_PATCHID \ /var/sadm/pkg/* \ pkginfo   cut -d -f2 \ sort -u	/bin/rpm -qa	Regedit export the branch: HKEY_LOCAL_MACHINE \ SOFTWARE
<b>User Limits</b>	ulimit -a /etc/security/limits	ulimit -a	ulimit -a	ulimit -a ulimit [-Ha] [-Sa]	N/A
<b>User Environ.</b>	set or env	set or env	set or env	set or env	set winmsd Environment Report msinfo32 Environment Variables
<b>System Logs</b>	/usr/bin/errpt -a	/var/adm/syslog \ /syslog.log	/var/adm/messages /var/log/syslog	/var/log/messages	Start>Programs> Administrative tools> Event Viewer
<b>System Admin</b>	/usr/bin/smit /usr/bin/smitty	/usr/sbin/sam	/etc/prtconf \ [-F] [-p] [-P] [-v] /usr/bin/admintool	/usr/sbin/setup /sbin/linuxconf Before 7.3: /usr/bin/redhat-config-* 7.3 and after: GUI	
<b>Compiler</b>	/usr/vac/bin/xlc	/usr/bin/cc	/opt/SUNWspro/bin/cc	/usr/bin/gcc	c++
<b>Debugger</b>	/usr/vac/bin/dbx	/opt/langtools/bin/xdb	/opt/SUNWspro/bin/dbx	/usr/bin/gdb	

## IDS Problem Determination Tutorial Series

### Combining IDS and OS Diagnostics

---

	/usr/bin/adb		/usr/bin/adb		
	/usr/bin/idebug		/usr/local/bin/gdb		
<b>Trace /</b>	/usr/bin/trace	q4*	pstack	/usr/bin/strace	
<b>Truss</b>		trace (HP 10.x)	/ucb/ps -auxww		
		tusc (HP 11.x)	/usr/sbin/snoop		
			/usr/bin/truss -p		
<b>Device</b>	/usr/sbin/lsvg	/usr/sbin/ldev	/usr/bin/devattr -	cat /proc/devices	
<b>Mgmt.</b>	/usr/sbin/lspv		v	/usr/bin/procinfo	
	/usr/sbin/ldev		prtconf -v	/usr/bin/ldev	
	-C				
	/usr/sbin/getlvc				
	b				
<b>Performanc</b>					
<b>e</b>					
<b>CPU</b>	/usr/bin/vmstat	/usr/bin/vmstat	/usr/bin/vmstat	/usr/bin/vmstat	Start>Run>Perfmon
	/usr/sbin/sar	/usr/sbin/sar	/usr/sbin/sar	/usr/bin/sar	
			/usr/bin/mpstat		
<b>I/O</b>	/usr/bin/iostat	/usr/bin/iostat -xt	/usr/bin/iostat -	/usr/bin/iostat -t	
			xcn		
<b>Memory</b>	vmstat	vmstat	vmstat	vmstat	
	/usr/bin/ipcs -	/usr/bin/ipcs -am	/usr/bin/ipcs -am	ipcs -am	
	am	swapinfo -t		/sbin/swapon -s	
<b>Processes</b>	lsps -a	/usr/bin/ps -elf	/usr/bin/ps avwx	ps -aelf	Task Bar>Right Click
	/usr/bin/ps aug	top	/usr/bin/ps -elf		Task Manager
	/usr/bin/ps -elf	glance			
<b>Network</b>	/usr/bin/netstat	/usr/bin/netstat	/usr/bin/netstat -	/bin/netstat	Run>Cmd>netstat
			ab		
*NOTE: iostat, vmstat, mpstat are usually followed by 2 arguments - the interval and number of iterations.					
<b>Startup/</b>	/etc/rc*	/sbin/init.d	/etc/rc.*	/etc/rc.d/init.d	Control Panel>Services
<b>Shutdown</b>	/etc/rc.d		directories	(scripts)	
<b>Scripts</b>	files for each		startup (S*)	/etc/rc.d/rc?.d	
	run		shutdown (K*)	(links to scripts)	
	level			startup(S*)	
				shutdown (K*)	
				/sbin/chkconfig	
				Text tool to	
				manage	
				rc scripts	
				/usr/bin/serviceco	
				nf	
				GUI tool to	
				manage	
				rc scripts	

.....

## ***Section 2 Diagnostic and Event Logs***

### **Section 2.1 Correlating IDS and System Events or Errors**

System messages and error logs are too often ignored. You can save hours, days, and even weeks on the time it takes to solve a problem if you take the time to perform one simple task at the initial stage of problem definition and investigation. That task is to compare entries in different logs and take note of any that appear to be related both in time and in terms of what resource the entries are referring to.

While not always relevant to problem diagnosis, in many cases the best clue is readily available in the system logs. If we can correlate a reported system problem with IDS errors we will have often identified what is directly causing the IDS symptom. Obvious examples are disk errors, network errors, and hardware errors. Not-so obvious are problems reported on different machines, for example domain controllers or NIS servers, which may affect connection time or authentication.

System logs can be investigated in order to assess stability, especially when problems are reported on brand new systems. Intermittent Assert Failures (traps) occurring in common applications may be a sign that there is an underlying hardware problem.

Here is some other information provided by system logs:

- Significant events such as when the system was rebooted
- Chronology of IDS Assert Failures (traps) on the system (and errors/traps/exceptions from other software that is failing)
- Kernel panics, out-of-filesystem-space, and out-of-swap-space errors (which may prevent the system from creating/forking a new process)

System logs can help to rule out crash entries in the IDS message log (online.log) as causes for concern. One consistent IDS crash recovery investigation was resolved by discovering that the system was rebooting - it turned out that the cleaning staff was unplugging the computer every morning at the same time!

If we see Physical (crash) recovery in the IDS online.log with no preceding errors, for example:

```
15:51:51 Informix Dynamic Server Initialized - Shared Memory  
Initialized
```

and, just before we see the following Windows system log entry:

```
11/11/02 3:51:51 PM EventLog Information None 6005
N/A SERVER1 The Event log service was started.
11/11/02 3:51:51 PM EventLog Information None 6009
N/A SERVER1 Microsoft (R) Windows NT (R) 4.0 1381 Service Pack 6
Multiprocessor Free.
```

Then, the IDS recovery was likely a result of a Windows shutdown.

This principle of correlating information extends to logs from any source and identifiable user symptoms. For example, it can be very useful to identify and document correlating entries from another application's log even if you can't fully interpret them.

## Section 2.2 Windows Event Logs and the Dr. Watson Log

Windows event logs can also provide useful information. While the system event log tends to be the most useful in the case of IDS crashes or other mysterious errors related to system resources, it is worthwhile obtaining all three types of event logs:

- System
- Application
- Security

To access event logs on Windows 2000 and Windows XP:

1. Select **Administrative Tools** from the Control Panel
  2. Select the **Event Viewer**.
- (or START/RUN/eventvwr.msc)

To access event logs on Windows NT:

Select **Start Menu -> Programs -> Administrative Tools -> Event Viewer**

From the event viewer, you can export event logs in two formats - in .evt format, which can be loaded back into an event viewer (for example on another machine) or in text format.

On NT, choose **Log** from the menu and **Save As**.

On Windows 2000, choose **Action** from the menu and **Save File As**.

Event viewer format is easy to work with since you can use the GUI to switch the chronology order, filter for certain events, and advance forwards or backwards.

Text files provide one significant advantage - you can trust the timestamps! When you export event logs in .evt format, the timestamps are in Coordinated Universal Time and get converted to the local time of the machine in the viewer. If you are not careful, you can miss key events because of time zone differences. Text files are also easier to search, but once you load an event log from another machine into the event viewer, it is easy enough to export it again in text format.

The Dr. Watson log, drwtsn32.log, is a chronology of all the exceptions that have occurred on the system. The IDS Assert Fail files are more useful than the Dr. Watson log, though it can be helpful in assessing overall system stability and as a document of the history of IDS Assert Failures (traps). The default path is:

`\Documents and Settings\All Users\Documents\DrWatson`

## Section 2.3 Avoiding Pitfalls and a First Glance at System Data

Once initial data has been collected, you may be able to find some information that signals a resource issue or some abnormal state with regards to the system, a process, or something IDS-specific. If you are not intimately familiar with the system and the activity taking place, you do need to be on your guard for some of the following pitfalls:

- The data might not have been captured during the time the problem occurred.
- Some of the data might point to an issue that is not of primary interest - in fact, there might be several different problems on the system that contribute to one or more symptoms of varying severities.
- The data might point to a secondary symptom. For example, a new condition or state is triggered from the primary problem that then causes a second symptom, and that symptom may overtake the first. A good example of this is when there is some resource contention on the database server such that requests get "tied up", and the application servers simply let more requests go to the server on previously idle connections. This might result in a secondary symptom as more and more memory is taken up on the server by active connections.

In all cases it is important to try to link up what the data implies with the problem as initially defined. Often this cannot be achieved immediately, but keeping this goal in sight can prevent you from wasting time trying to solve the wrong problem, or worse, one that doesn't exist.

On UNIX, problem investigation often starts with the following:

- Check vmstat or iostat output for the amount of system and user CPU being consumed versus wait and idle time. Wait time usually, but not always, means waiting on I/O. If system CPU is equal to or greater than user CPU, this almost always indicates a problem.
- Check the top CPU-using processes by ordering ps output by descending CPU penalty (C) and memory usage (RSS and SZ columns). For example:  

```
ps -elf | sort +5 -rn
```
- Are the same processes near the top? Is the CPU penalty staying in the 60-120 range? This would mean these processes are constantly consuming CPU cycles and need to be investigated.

On Windows, check for the top CPU and memory users by ordering the output on those columns.

## Section 2.4 Using perfmon on Windows

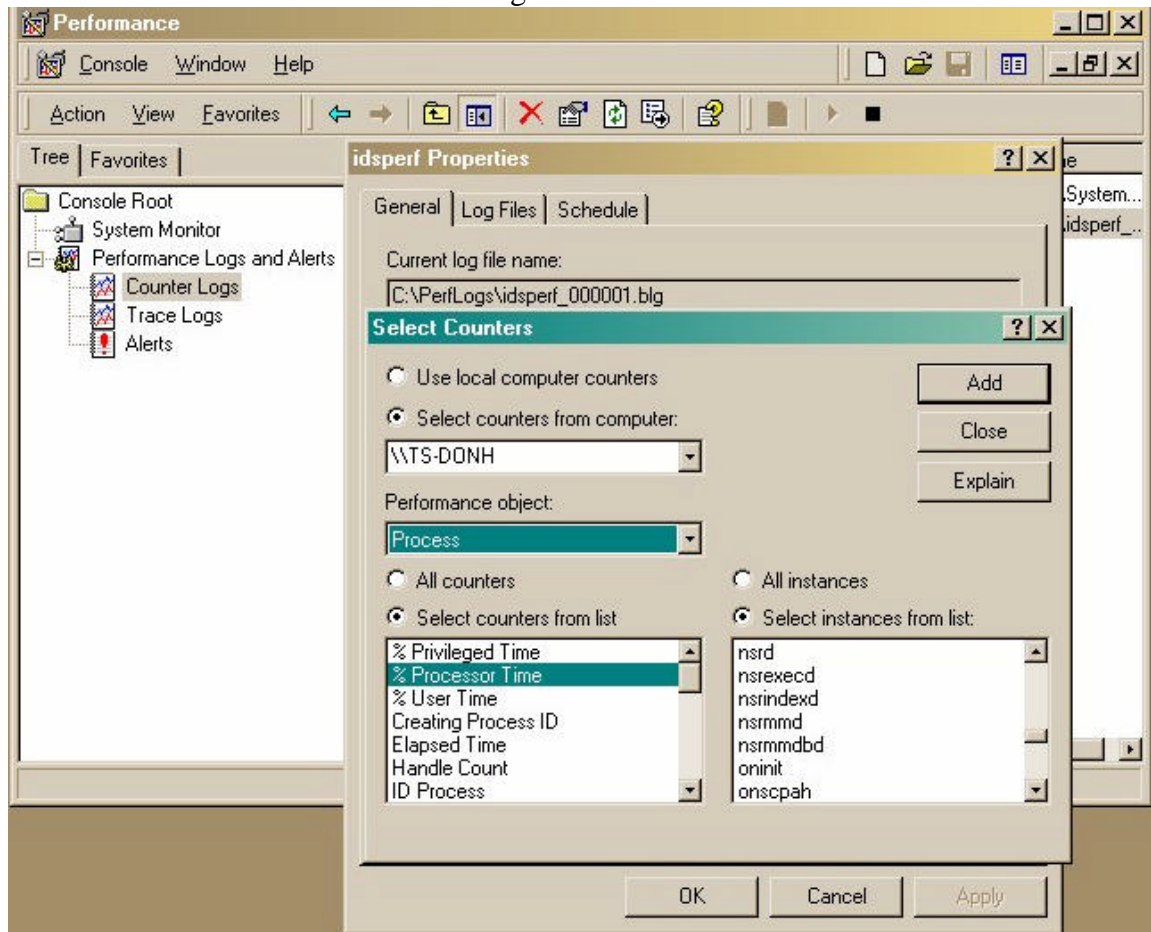
The **perfmon** tool, shipped with Windows, is used to capture performance data and statistics on resource usage. Understanding how to set up and capture a perfmon log is crucial to many types of problem investigation.

The first catch is that, for monitoring I/O, disk counters need to be enabled by running **diskperf -y** ( -ye for stripe sets), followed by a reboot.

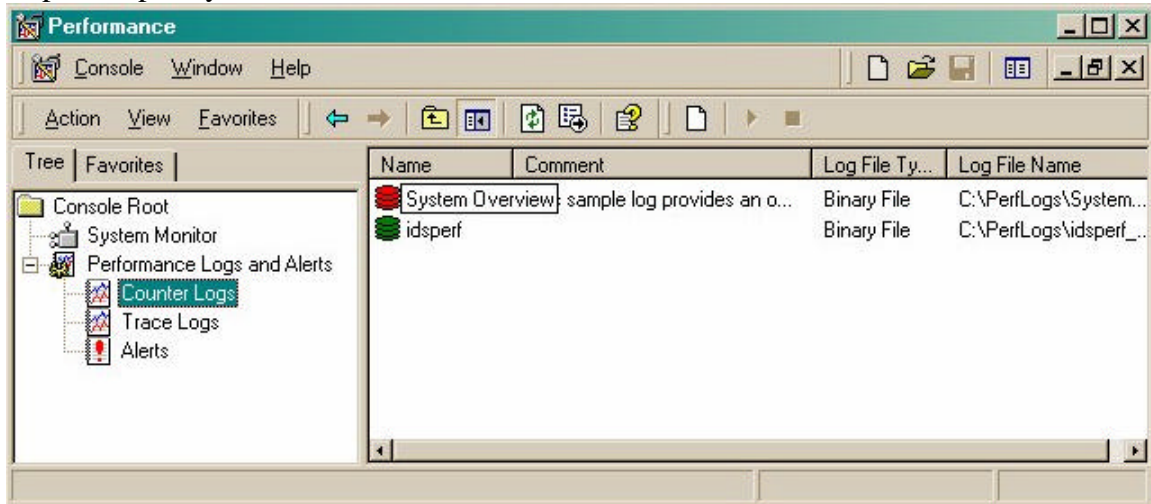
To set up a log on Windows 2000:

1. Run perfmon from a command prompt.
2. Select **Performance Logs and Alerts** from the left frame.
3. Right-click on **Counter logs** and select **New Log Settings**.
4. Give the settings a name and click on **Add** to add some performance counters.

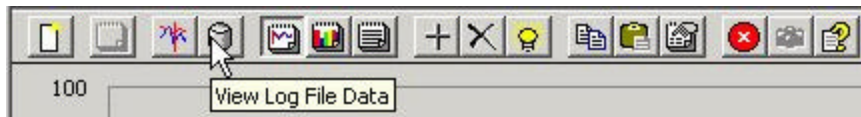
Your screen should now look something like this:



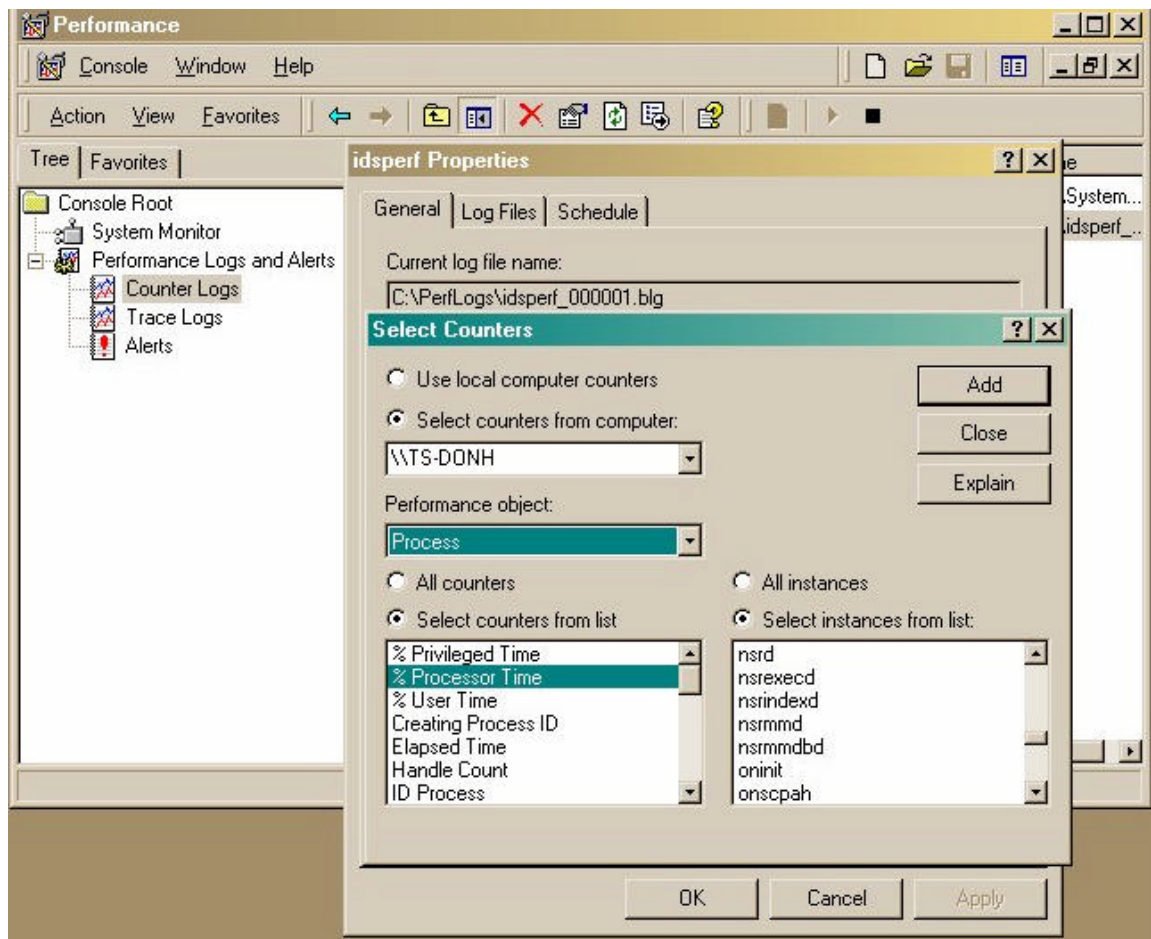
Now close the windows by clicking on the appropriate **Add**, **Close**, and **OK** buttons. Once you are back to the initial screen, double-click on **Counter Logs**. You may see that the log is already running, as indicated by the green icon to the left of the log name. If it is not started, right-click on the log and select **Start**. For example, if your log is called idsp perf idsp perf, your screen should now look like this:



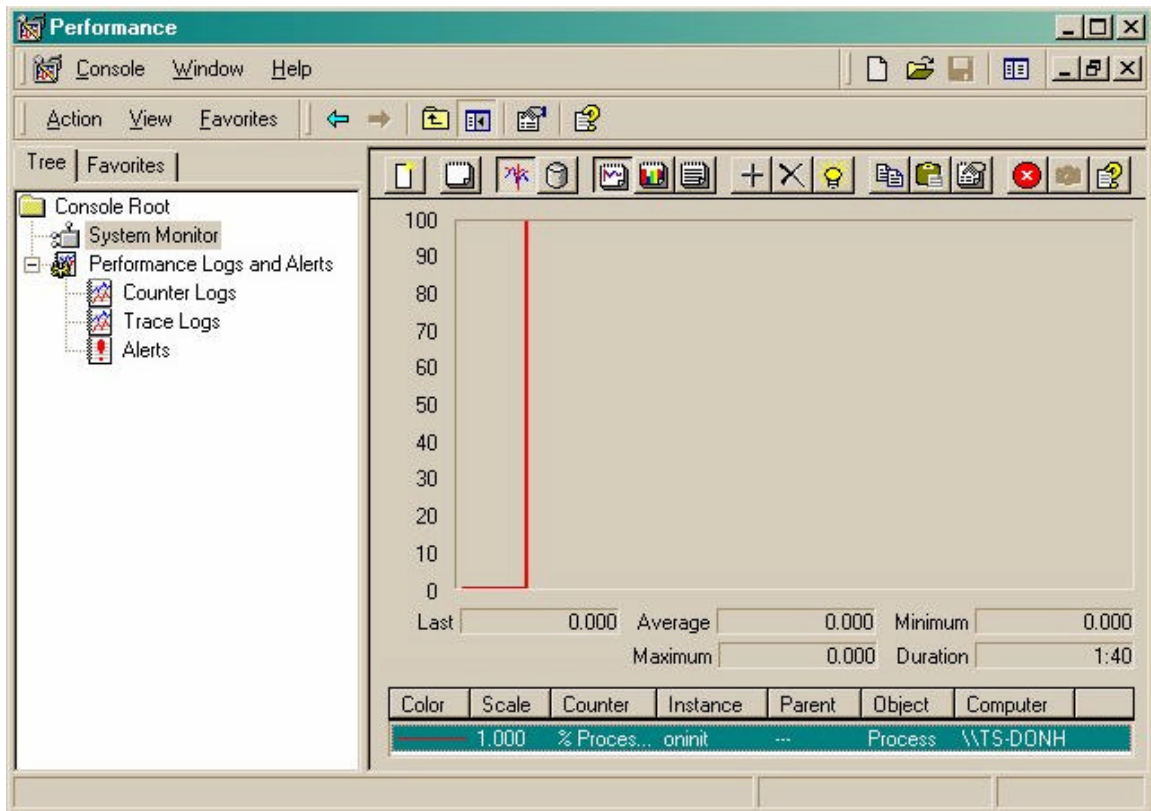
To view the log file, select **System Monitor** from the tree, then select the **View Log File Data** icon:



Once you have selected a log file, you still need to add counters to the chart. Right-click in the chart area, and add counters as desired. Here is a sample setting and output where **All instances** (processes) of the **% Processor Time** counter have been selected. Setting:



To make the output easier to digest, select an instance (record) and highlight it in white by pressing Ctrl+H, then cycle through the list and delete the instances that aren't interesting (those that show low CPU utilization):



One of the attractive features of perfmon is the ability to combine IDS and system performance counters together in one log.

## **Section 3 Operating System Error Examples on UNIX**

**Note:** All the exercises in this section were written and tested on the Linux platform. Many of them will work on other platforms but some may need to be adapted.

### **Section 3.1 File Permissions and Space Issues**

It is not uncommon to run into permission or space issues, especially when setting up an instance for the first time, or when adding space to an instance. The following examples guide you through two scenarios where you will be faced with space and permission issues. These examples assume you already have an instance created, the instance is online, and you are logged in as user “informix”.

#### **Exercise:**

1. Create a directory in /tmp in which to test

```
cd /tmp
mkdir test
cd test
```
2. You will need a filesystem with limited space for this example. Run the following steps as user “root” to create a small filesystem within a filesystem on Linux. This small filesystem will be created in /tmp/test.

```
mkdir mnt
dd if=/dev/zero of=my.fs bs=1k count=2000
mkfs -t ext2 my.fs
mount -o loop -t ext2 my.fs mnt
chmod 777 mnt
```
3. Log back in as user “informix” and touch a file into which we’ll create a new chunk and dbspace.

```
cd mnt
touch mydbs.001
```
4. Change the permissions to read-only

```
chmod 444 mydbs.001
```
5. Try to create the dbspace

```
onspaces -c -d mydbs -p /tmp/test/mnt/mydbs.001 -s 5000 -o 0
```
6. You should have seen the following messages

```
Verifying physical disk space, please wait ...
The chunk '/tmp/test/mnt/mydbs.001' must have READ/WRITE permissions
for owner and group.
```
7. The reason for this error is because IDS chunks require permissions of 660, or read/write for owner and group as the message indicates. Checking the permissions using

the “ls” command will show that the “mydbs.001” file has only read permissions for owner, group, and world.

```
ls -al
total 17
drwxrwxrwx 3 root root 1024 May 14 16:30 .
drwxrwxr-x 3 informix informix 4096 May 14 16:26 ..
drwxr-xr-x 2 root root 12288 May 14 16:27 lost+found
-r--r--r-- 1 informix informix 0 May 14 17:22 mydbs.001
```

8. We need to fix the file permissions so “mydbs.001” has read/write permissions for owner and group.

```
chmod 660 mydbs.001

ls -al
total 17
drwxrwxrwx 3 root root 1024 May 14 16:30 .
drwxrwxr-x 3 informix informix 4096 May 14 16:26 ..
drwxr-xr-x 2 root root 12288 May 14 16:27 lost+found
-rw-rw---- 1 informix informix 0 May 14 17:22 mydbs.001
```

9. Now, we will test a space limitation problem. The first idea would be to set ulimit to restrict the size of created files, however, on Linux, onspaces does not obey ulimit. This is why we created our own filesystem of limited size. Try creating the space again.

```
onspaces -c -d mydbs -p /tmp/test/mnt/mydbs.001 -s 5000 -o 0
```

10. This should produce the following message

```
Verifying physical disk space, please wait ...
/tmp: write failed, file system is full.
```

11. The reason for this error is our filesystem was created with a size of 2 MB and the dbspace/chunk we are trying to create is 5 MB’s in size. We can verify how much space is free in our filesystem by using the “df” command.

```
df -k
Filesystem 1k-blocks Used Available Use% Mounted on
/dev/hda1 9582284 2639252 6456268 30% /
/dev/hdb1 4917648 4249428 418412 92% /opt
/tmp/test/my.fs 1963 13 1850 1% /tmp/test/mnt
```

The “-k” option indicates that sizes displayed in the output should be in kilobytes. We see that we have 1850 KB currently available in “/tmp/test/mnt”.

12. To actually create the space, simply reduce to size of the chunk to 1 MB instead of the 5 MB size we’ve been specifying.

```
onspaces -c -d mydbs -p /tmp/test/mydbs.001 -s 1000 -o 0
```

13. Success! We should see the following

```
Verifying physical disk space, please wait...  
Space successfully added.
```

```
** WARNING ** A level 0 archive of Root DBSpace will need to be  
done.
```

14. This concludes this portion of the tutorial, so let's clean up after ourselves

```
onospace -d mydbs  
ontape -s -L 0  
cd ..
```

```
(as user root)
```

```
umount mnt  
rm -rf mnt  
rm mt.fs
```

## Section 3.2 Connectivity Errors: 951, 952, 956 25588, 25582

Connectivity errors happen most often when connecting to a database from another database or client application, however, they can also occur when from a database server to itself. The following tutorials assume you already have an instance created, the instance is online, the instance has a tcp/ip connection configured, and you are logged in as user "informix".

The first error we will look at is a 951 error. 951 errors typically occur because your username is not recognized (at the OS level) on the system where the database server is installed.

### Exercise:

1. The 951 error is simple to produce. The easiest way is to go into dbaccess  
dbaccess
2. Then from the menu, choose "Connection"
3. And then choose "Connect"
4. You will be presented with a choice of available connections found in your sqlhosts file. Chose your tcp/ip connection name.
5. When prompted for a username, enter one that does not exist on your system
6. Do the same for password
7. At the bottom of your dbaccess window, you will see something similar to:

```
951: Incorrect password or user bob is not known on the database
server.
```

8. To begin troubleshooting this error, make sure you can log into to Unix or Windows system as the user you are trying to connect to the database as.

The 952 error is similar, but instead of an invalid username it is caused by an invalid password.

**Exercise:**

1. Follow the steps above, but when prompted for a username, enter a valid username. When prompted for a password, enter an invalid password.

2. You should see the following error:

```
952: User (jousley)'s password is not correct for the database
server.
```

3. Troubleshoot this the same way as you would step 8 above. Pay special attention to the password.

956 errors happen when the remote host (database server) you are trying to connect does not trust the client (local host) making the connection. This is slightly more difficult to show in an example as it requires two hosts. The following example requires the IDS product installed on two servers. You'll need one database server configured and running (probably the one used for the prior examples). We'll call this server, "A". We need the dbaccess component from the other installation on the second server and an sqlhosts file configured to connect to the tcp/ip connection of the first server. We'll call this server, "B".

**Exercise:**

1. Make sure that B is not already trusted by A by looking in the file "/etc/hosts.equiv" on A. You should not see B's hostname or ip address listed in this file. Also, make sure there is not a "+" sign at the top of the file.

2. From machine B, login in as user "informix", and set your INFORMIXDIR environment variable to the tcp/ip connection to the instance on machine A. (All shell commands assume the bash shell.) For example:

```
export INFORMIXSERVER=iif930uc2_tcp
```

3. Now try to make a connection to a known database in the instance on machine A using dbaccess.

```
dbaccess sysmaster -
```

4. You should see the following error:

```
956: Client host or user (informix@webspot.lenexa.ibm.com) is not
trusted by the server.
```

No such file or directory

5. To fix the problem we need machine A to trust machine B. To do this add an entry to the end of /etc/hosts.equiv with the fully qualified hostname of machine B. You will need to create ans/or edit this file as user "root". For example:

webspot.lenexa.ibm.com

6. Now try making the connection from machine B again.

dbaccess sysmaster -

7. You should now see the following indicating a successful connection:

Database selected.

8. You can hit "break" (control-c on most Linux systems) to end this dbaccess session.

25588 errors typically occur when there is a problem connecting to a shared memory connection. The previous examples have used a tcp/ip connection. For the next example we will use a shared memory connection on for the instance on machine A. We will no longer need machine B.

#### **Exercise:**

1. Make sure the instance is online and we have a shared memory connection configured in both the sqlhosts file and the onconfig file.

export INFORMIXSERVER=iif930uc2\_shm

2. We need to change our working sqlhosts entry for our shared memory connection to something invalid. This is what I have configured:

iif930uc2\_shm onipcshm iknow iif930uc2\_shm

I changed it to this:

iif930uc2\_shm onipcshm iknow broken

3. Try making a connection to a known database via dbaccess.

dbaccess sysmaster -

4. You should see the following error:

25588: The appl process cannot connect to the database server  
iif930uc2\_shm.

No such file or directory

5. Correct the sqlhosts file entry:

iif930uc2\_shm onipcshm iknow iif930uc2\_shm

6. Try the connection again.

dbaccess sysmaster -

7. You should successfully connect and see:  
`Database selected.`
8. You can hit “break” (control-c on most Linux systems) to end this dbaccess session.
9. As a side note, another simple way to reproduce a 25588 error is to simply bring down the database server with “onmode -ky”, then try your dbaccess connection again.

25582 typically show up when an existing network connection is interrupted. For this example we will once again use our tcp/ip connection. You will need two telnet sessions for this example.

**Exercise:**

1. Make sure the instance is online and your INFORMIXSERVER environment variable is set to the tcp/ip connection.

```
export INFORMIXSERVER=iif930uc2_tcp
```

2. In the first telnet window, initiate a dbaccess connection to a known database  
`dbaccess sysmaster -`
3. This connection should be successful and you will see the following:  
`Database selected.`
4. In the second window, take the instance offline:  
`onmode -ky`
5. In the first window, try to run a query:  
`> select * from systables;`
6. This will cause the 25582. You should see the message below and your session will appear hung.  
`25582: Network connection is broken.`
7. In the second window, bring the instance back up  
`oninit`
8. This should release the first window and you will see something similar to:  
`Error in line 1  
Near character position 1  
>`

### Section 3.3 Shared Memory Related Issues

Shared memory issues are very common since an IDS instance cannot live without several types of shared memory. Common problems arise when an instance tries to grab a

portion of shared memory that is already in use, or when there simply is no more room to allocate shared memory. We will look at some common situations that cause shared memory errors

**Exercise:**

1. First we will simulate the occurrence of two instance trying to use the same shared memory space, this is extremely easy to do. Start by bringing an instance online, or making sure it is online.

```
onstat -  
  
shared memory not initialized for INFORMIXSERVER 'iif930uc2_shm'  
  
oninit  
onstat -
```

```
Informix Dynamic Server Version 9.30.UC2 -- On-Line -- Up 00:00:23 --  
28284 Kbytes
```

2. Then, simply try to bring the engine up again (kind of like trying to start your car when it's already started only it doesn't make that horrible grinding noise).

```
oninit
```

3. You should see the following at the command line:

```
oninit: Fatal error in shared memory creation
```

4. If you peek into the message log (online.log), you should see something similar to:

```
onstat -m  
  
Informix Dynamic Server Version 9.30.UC2 -- On-Line -- Up  
00:04:32 -- 28284 Kbytes  
  
Message Log File: /opt/informix/logs/iif930uc2/online.log  
12:28:42 Informix Dynamic Server Started.  
12:28:42 shmget: [EEXIST][17]: key 52574801: shared memory  
already exists  
12:28:42 mt_shm_init: can't create resident segment
```

You can see that we get errors indicating that we already have those areas of shared memory allocated and we cannot allocate them again.

Another type of failure can result when we try to grab more shared memory segments than the kernel will allow. On linux, this is easy to test, but please make sure to only do this on a test system, not a live production system.

**Exercise:**

1. Open two telnet sessions, in the first, login as user "informix", in the second, login as user root.

2. In the first telnet session, as user “informix”, Bring your instance offline if it already isn’t:

```
onmode -ky
```

3. In the second session, as user “root”, run ipcs to determine the number of shared memory segments currently allocated.

```
ipcs -mu
```

4. You should see something like this:

```
----- Shared Memory Status -----  
segments allocated 5  
pages allocated 1340  
pages resident 41  
pages swapped 263  
Swap performance: 0 attempts 0 successes
```

5. This shows that we currently have five segments in use. Now set the “shmmni” kernel parameter to equal this value. On linux, the kernel parameter values are compiled into the kernel. However, there is a subsystem that store the current values, and through this system, you can dynamically change parameters. The changes take affect immediately. Change to the “/proc/sys/kernel” subdirectory.

```
cd /proc/sys/kernel
```

6. Examine the current value of “shmmni” on your system.

```
cat shmmni
```

7. It typically defaults to 4096 on a newly configured system. You should see something such as the following returned:

```
4096
```

8. Now, lets change this value to the number of segments we already have allocated so that no more can be created.

```
echo 5 > shmmni
```

9. In your first telnet window, as user “informix”, try to bring up your instance.

```
oninit
```

10. This should reproduce the following error:

```
oninit: Fatal error in initializing ASF with 'ASF_INIT_DATA' flags;  
asfcode = '-25580'.
```

11. Examine the message log (online.log).

```
onstat -m
```

12. You should find something similar to this:

```
Thu May 22 13:02:08 2003
```

```
13:02:08 Event alarms enabled. ALARMPROG =  
'/opt/informix/iif930uc2/etc/no_log.sh'  
13:02:08 Booting Language from module <>  
13:02:08 Loading Module  
13:02:08 Booting Language from module <>  
13:02:08 Loading Module  
13:02:13 shmget: [ENOSPC][28]: key 52574803: out of shmid's, check  
system SHMMNI  
13:02:13 (3) shm creation of shmem segment failed
```

13. This is because we cannot allocate any more shared memory segments.
14. In the second window, as “root”, put back the original “shmmni” value.  

```
echo 4096 > shmmni
```
15. In the first window, try bringing up the instance  

```
oninit
```
16. This should be successful.

## ***Section 4 Defining a Performance Problem and Initial Data Collection***

### **Section 4.1 Clarifying the Problem**

Performance problems cover a wide range of scenarios:

- Identifiable query performing slower than expected
- Workload or batch job not completing as soon as expected, reduction in transaction rate or throughput
- Overall system slowdown
- Suspected bottleneck in some type of system resource such as CPU, I/O, memory and network.
- Query or workload consuming more resource than expected or available
- Comparison is being made between one system and another
- Query, application, IDS engines, or system hangs

There are some subtleties in the scenarios depicted above. For problem diagnosis purposes, it is important to clarify whether something is not meeting expectations or is exceeding resource capacity. Sometimes it is both.

For problem determination purposes, hangs can be lumped together with performance problems because many investigative strategies apply to both. In addition, it may not be possible at first to define the problem as a hang versus a performance problem. To a user waiting for a response, a long-running job can look like a hang even if in fact much activity can be taking place on behalf of the application on the database server. There can also be a significant buildup of activity during a severe system slowdown such that all or most commands appear to hang on a system.

In addition to characterizing the problem correctly in terms of where the symptom is observed (query/application/system resource) and what is wrong with it (slowness or too much resource used), you require many other pieces of information to put the problem in context:

- When the problem started occurring and what recent changes were identified, if any (hardware or software upgrades, new application rollout, tuning, additional users, etc.)
- How often it occurs
- Exactly what is observed and by whom
- What steps have been taken so far in terms of monitoring, tuning, and other changes
- What are the requirements - as detailed as possible
- What benchmarks have been taken
- What other problems, if any, have occurred on the system
- Any diagnostic information that was gathered with regards to IDS engines and OS

## Section 4.2 IDS diagnostics that can be performed quickly

The onstat utility is a tool that assists in quickly narrowing down problems. Unless the problem is already narrowed down to a specific application or query, it is best to start with iterations (at least 2, preferably 3) of the more general snapshots:

1. onstat -a
2. onstat -g all
3. onstat -g stk all

When working towards defining a problem, it is best to start with the online.log, onconfig file and onstats to get a feel for the type of activity taking place: how many processes are running in the server, how many connections exist, whether lock issues are occurring, overall buffer hit ratios and I/O time, sort activity, etc.

It is often required to script some ongoing database monitoring. Here is an example of such a script, where the iostat and vmstat timings control the frequency of the loop:

```
while [ 1 ] do

    datestamp=`date +%m%d"`. `date +%H%M" `

    onstat -a > ons.a.$datestamp

    onstat -g all > g.all.$datestamp

    ... etc. for each desired type of snapshot.

    ps -elf | sort +5 -rn > pself.$datestamp

    ps aux | sort +5 -rn > psaux.$datestamp

    ipcs -a > ipcs.$datestamp

    uptime >> vmstat.out

    vmstat 1 11 >> vmstat.out

    echo " " >> vmstat.out

    uptime > iostat.$datestamp

    iostat 30 16 >> iostat.$datestamp
done
```

For example, if you experience an nsf.lock issue, you may want to start gathering stacks for holder and waiter threads.

1. onstat -g lmx to identify the holder and waiters of the mutex.
2. onstat -g stk <thread id> of the thread that's holding the mutex and the threads that are holding

3. onstat -g nta
4. onstat -g act
5. pstack
6. truss

You can also use a script to gather onstat data.

```
#!/usr/bin/ksh # Informix profile to source the env

# Log file definitions

LOGPATH=`pwd`

STAMP=`date +%Y%m%d%H%M%S`

LOG=$LOGPATH/nsfmutex.$STAMP

date >> $LOG

for i in 1 2 3 4 5

do

echo "*****Holder thread *****" >> $LOG

onstat -g lmx | head -20 >> $LOG

ltid=`onstat -g lmx | grep nsf.lock | awk '{print $4}'`

onstat -g stk $ltid >> $LOG

# If -g stk does not give stack info, please use pstack

#pstack {vpid} >> $LOG

echo "*****Waiter threads *****" >> $LOG

wtid=`onstat -g lmx | grep nsf.lock | awk '{print $6}'`

for i in $wtid

do

if [ "x$i" != "x" ]

then

onstat -g stk $wtid >> $LOG

# If -g stk does not give stack info, please use pstack

#pstack {vpid} >> $LOG
```

```
fi  
  
done  
  
sleep 2  
  
done
```

## Section 4.3 IDS function call stacks, trap files, and tracing

### Assert Failure Stack Format

An Assert Failure is a type of system crash, failure or a warning that may or may not be able to correct itself. When there is an assert failure, there will be a file dumped on a directory specified under DUMPDIR config parameter or in /tmp by default. The name of the file has the format “af.xxxxx”.

An Assert Failure will show the session and thread that caused an af and the results and action to take. When you go down further, you see the stack for the thread. The stack is a chunk of memory used by a given process. It shows the function that caused an af. The Function that is below the af signal handling code is the one that causes an af.

Below is a snippet of an af file.

```
00:31:01  
  
00:31:01 Informix Dynamic Server Version 9.30.UC3 Software Serial Number  
AAD#J328675  
  
00:31:01 Assert Failed: errors occurred during mt_notifyvp  
  
00:31:01 Who: Session(114407, root@nelachostlige, 25910, 0)  
  
Thread(8, soctcplst, 0, 1)  
  
File: mt.c Line: 10787  
  
00:31:01 Stack for thread: 8 soctcplst  
  
base: 0x383a6000  
  
len: 36864  
  
pc: 0x085595d0  
  
tos: 0x383ae050  
  
state: running
```

```
vp: 1

0x085595d0 (oninit)afstack (0x3833aa40, 0x878ae20, 0xd3b, 0x383ae108, 0x0, 0x752f)

0x08558c6f (oninit)afhandler(0x3, 0x86fb3c0, 0x0, 0x0, 0x601, 0x1)

0x0855871a (oninit)afcrash_interface(0x86fb3c0, 0x0, 0x0, 0x8785c94, 0x2a23,
0x401f9000)

0x0853e37a (oninit)notifyvp(0x1, 0x0, 0x0, 0x383ae6ac, 0x854bf90, 0x1)

0x0853e45b (oninit)mt_notifyvp(0x1, 0x0, 0x2000000, 0x0, 0x2, 0x401f9000)

0x0854bf90 (oninit)mt_shm_create_segment(0x2000000, 0x0, 0x2, 0x0, 0x37d08e08,
0x21000)

0x0854fc2d (oninit)morecore(0x2, 0x21000, 0xffffffff, 0xffffffff, 0x383ae6f8,
0x387b8338)

0x08550e49 (oninit)mt_shm_get_blk(0x37d08e08, 0x21000, 0x387b81f0, 0x382642d0,
0x383ae730, 0x853756c)

0x08553ea1 (oninit)mt_alloc_stack(0x20000, 0x3851ba38, 0x3fa8fd70, 0x830eee0,
0x852b163, 0x383ae740)

0x0853756c (oninit)create_tcb(0x20000, 0x382642d0, 0x383ae85c, 0x8545896,
0x830eee0, 0x0)

0x0853a7e2 (oninit)mt_fork (0x830eee0, 0x0, 0x0, 0x2, 0x20000, 0x37e398a8)

0x08545896 (oninit)activate_session(0x3fa8fd70, 0x830eee0, 0x0, 0x382642d0, 0x0,
0x383aeba8)

0x08571f46 (oninit)spawn_thread(0x383aeelc, 0x383aec54, 0x383aeb4, 0x383aeba8,
0x3fa8fd70, 0x383a2b28)

0x08571806 (oninit)sql_listener(0x0, 0x0, 0x0, 0x0, 0x0, 0x2e)

0x08541c59 (oninit)startup (0x2e, 0x383af060, 0x383a2fc0, 0x60, 0x70752081, 0x0)

0x00000000 (*nosymtab*)0x0

00:31:01 See Also: /tmp/af.3f0f4c4, shmem.3f0f4c4.0
```

The above snippet tells you that this stack has a base address of 0x383a6000 and a length of 36864. The top of the stack is 0x383ae050. The program counter is 0x085595d0. Stack shows that we started from sql\_listener and when we go up further we would find assert failure functions like afstack(), afhandler() and afcrash\_interface(). The function that is below these AF functions, is the one that caused an af. Here we failed on notifyvp() from mt\_shm\_create\_segment() function.

You can also dump the raw memory of the stack using `onstat -g dmp 0x383a6000 36864`

### Causing an AF stack

Other information that can be useful includes an IDS stacktrace which shows the functions that are executing. Often the function names alone provide some indication of what the process is doing.

For a problem where most IDS commands are hanging but there is still a lot of activity taking place on the server, take a stacktrace of the thread that is hanging by running the following.

```
onstat -g stk {thread-id} or onstat -g stk
```

In the case where there is clearly one application or processor that is not responding, you can take trace and call stack diagnostics specific to that process or group of processes. Follow the table below to generate IDS diagnostic trap files on an individual process. (If a process is in a tight loop, it may not pass through the function that tells it whether or not to start writing trace records). In this scenario, you will need to contact support for other possible ways to generate a stack trace. While they could be listed here, there may be unexpected consequences that is beyond the scope of this document.

To dump the stack from 9.21 engines when 'onstat -g stk' does not work, you can run `onstat -g ath` and note the vp that the thread is running on

```
onmode -X stack VP
```

or

```
onmode -X s VP
```

(onmode -X is an undocumented option)

For example, `onmode -X stack 3` will request a stack dump for the current thread on VP #3.

The 'onmode -X stack VP' command will create an assert failure file and dump the stack trace into the assert failure file. For path and assert failure file name check the online log file.

The following is an online.log output of 'onmode -X stack VP' command:

```
12:32:18 stack trace for pid 912 written to /tmp/af.3f25921
```

**Note :** If `onmode -X stack` is run against ADMVP on 9.21 and 9.3, the engine will crash. Please refer bug 152839 for more details.

### **Onmode -I Command**

If there is an error number associated with the performance problem, you can use onmode -I command to specify one or more error numbers to produce an assertion failure. Also, if the environment variable AFDEBUG is set at the time of the error, it will deliberately hang the engine.

For example, if you get error 201, "A Syntax error has occurred", use onmode -I 201 which will set an error trap.

```
17:18:22 TrapError set, errno = 201, session_id = -1
```

When there is an error 201, the engine would dump an af file under DUMPDIR config parameter. You can use onmode -I to clear this error trap and can be verified from the online.log

### **Engine Tracing Utilities**

There are some tracing utilities available and included with the engine. But using them can have unexpected consequences and as such should only be used under the advice of IBM.

#### **Avoiding pitfalls and a first glance at system data**

Once initial data has been collected, you may be able to find some information that signals a resource issue or some abnormal state with regards to the system, a process, or something Informix-specific. If you are not intimately familiar with the system and the activity taking place, you do need to be on your guard for some of the following pitfalls:

- The data might not have been captured during the time the problem occurred.
- Some of the data might point to an issue that is not of primary interest - in fact, there might be several different problems on the system that contribute to one or more symptoms of varying severities.
- The data might point to a secondary symptom. For example, a new condition or state is triggered from the primary problem that then causes a second symptom, and that symptom may overtake the first. A good example of this is when there is some resource contention on the database server such that requests get " ;tied up", and the application servers simply let more requests go to the server on previously idle connections. This might result in a secondary symptom as more and more memory is taken up on the server by active connections.

In all cases it is important to try to link up what the data implies with the problem as initially defined. Often this cannot be achieved immediately, but keeping this goal in sight can prevent you from wasting time trying to solve the wrong problem, or worse, one that doesn't exist.

On UNIX, problem investigation often starts with the following:

- Check vmstat or iostat output for the amount of system and user CPU being consumed versus wait and idle time. Wait time usually, but not always, means waiting on I/O. If system CPU is equal to or greater than user CPU, this almost always indicates a problem.
- Check the top CPU-using processes by ordering ps output by descending CPU penalty (C) and memory usage (RSS and SZ columns). For example:  
`ps -elf | sort +5 -rn`
- Are the same processes near the top? Is the CPU penalty staying in the 60-120 range? This would mean these processes are constantly consuming CPU cycles and need to be investigated.

On Windows, check for the top CPU and memory users by ordering the output on those columns. I/O is more difficult to judge without getting perfmon output.

## Section 4.4 OS tracing and debugging utilities

### Tracing

On any OS, we can trace system calls and signals for a process using its process-id(pid). Here is the command for all OS.

	AIX	HP-UX	Solaris	Linux
Trace	/usr/bin/trace	/usr/bin/trace	/usr/bin/	/usr/bin/strace
		(freeware)	truss -p	/usr/bin/ltrace
	/usr/bin/smit		pid	

### Debuggers

On any OS, you can either use gdb or dbx to attach a process to get a stack. This is also another way of getting a stack for a process.

	AIX	HP-UX	Solaris	Linux
Debugger	/usr/vac/bin/dbx	/opt/langtools/	/opt/SUNWs	/usr/bin/gdb
	/usr/bin/adb	bin\	pro/bin/dbx	
	/usr/bin/idebug	/xdb	/usr/bin/adb	
			/usr/local/bi	
			n/gdb	

## ***Summary and feedback***

### **What you should know**

Of course this is only an introduction to get you started.

You can solve some problems or at least get on the right track just by taking an initial look at the right data. Other problems are more complicated or subtle and require a strategic approach that uses several data collection and more complex analysis techniques.

Here are a few additional resources available:

- IDS online documentation: <http://www-3.ibm.com/software/data/informix/pubs/>
- UNIX man pages contain a wealth of information about tools
- AIX documentation: [publibn.boulder.ibm.com/cgi-bin/ds\\_form?lang=en\\_US&viewset=AIX](http://publibn.boulder.ibm.com/cgi-bin/ds_form?lang=en_US&viewset=AIX)
- Solaris documentation: [docs.sun.com/](http://docs.sun.com/)
- HP-UX documentation: [docs.hp.com/](http://docs.hp.com/)
- Linux documentation: [docs.linux.com/](http://docs.linux.com/)
- Windows Support: [microsoft.com/technet/support](http://microsoft.com/technet/support)